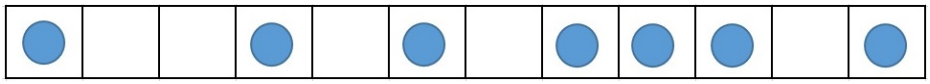
3. A_Three – Penny Dreadful

Program Name: A_Three.java **Input File: a_three.dat**

A simple penny game involves a strip of several adjacent spaces, on which pennies are randomly placed. The number of squares S (2<S<100) is random, as are the number of pennies P placed, as long as P (2<=P<S) is less than S.

The rules for a complete game, should you wish to play it, are as follows:

- After the pennies are randomly placed by one player, the players take turns moving pennies to the left.
- A move can be 1, 2, 3, 4 or 5 spaces, as long as
  - a penny does not share the same space as another penny after the move, and
  - a penny can't pass another penny during the move.
- The winner of the game is the last player to move a penny.



For example, on the game board shown above, with 12 squares, and 7 pennies at positions 1, 4, 6, 8, 9, 10 and 12, 4 pennies can move 1 space (the pennies at positions 4, 6, 8 and 12), 1 penny can move 2 spaces (the one in position 4), but none can move any further.

Your job is to report the possible number of moves by the first play, itemized by possible number of spaces moved. The output for this situation would be: 4 1 0 0 0, which means there are 4 possible 1 space moves, 1 possible 2 space move, and no possible moves of 3, 4, or 5 spaces.

Input: A data file with several sets of data. Each data set is a series of integers, all on one line, with single space separation. The first integer S indicates the number of squares on the penny strip, the second value P shows the number of pennies on the strip, followed by P positive values indicating the penny positions on the board. Each of the penny positions are guaranteed to be unique, in ascending order, and less than or equal to S. P will always be at least 2, and less than S.

Output: Five values per data set, the first indicating the number of possible 1 space moves by the pieces on the board, the second the number of 2 space moves, and so on…3 space moves, 4 space moves, and 5 space moves.

Sample input:

```
12 7 1 4 6 8 9 10 12
14 6 1 3 6 7 10 14
10 2 4 9
```

Sample output:

```
4 1 0 0 0
4 3 1 0 0
2 2 2 1 0
```